10-23-00

A

Please type a plus sign (+) inside this box ⇒ [+]

PTO/SB/29 (1/98)
Approved for use through 09/30/00. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1 53(b))		
	Attorney Docket No.	M-6017-1C US
	First Named Inventor or Application Identifier	Kenneth J. Klask
	Title	Graphical User Interface Engine For Embedded Systems
	Express Mail Label No.	EL 702 144 275 US

APPLICATION ELEMENTS
See MPEP chapter 600 concerning utility patent application contents

ADDRESS TO: Commissioner for Patents
Box Patent Application
Washington, D.C. 20231

1. ☒ Fee Transmittal Form - ***see page 2 of this form.***
(Submit an original, and a duplicate for fee processing)
2. Application:
☒ Specification: (preferred arrangement set forth below)
Descriptive title of the Invention,
Cross References to Related Applications,
Reference to Microfiche Appendix,
Background of the Invention,
Brief Summary of the Invention,
Brief Description of the Drawings, and
Detailed Description (all totaling 25 pages)
Appendix(ces) ___,___, & ___ (___pages)
☒ Claim(s) 6 pages
☒ Abstract of the Disclosure 1 page
3. ☒ Drawing(s) (35 USC 113) [Total Sheets 4]
4. Oath or Declaration ☐ unsigned [Total Pages 2]
a. ☐ Newly executed (original or copy)
b. ☒ Copy from prior application (37 CFR §1.63(d))
(for continuation/divisional with Box 17 completed)
c. ☐ DELETION OF INVENTOR(S)
Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1 63(d)(2) and 1 33(b)
5. ☒ Incorporation By Reference (useable if Box 4b is checked)
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☐ Microfiche Computer Program Appendix
consisting of ___ pages of microfiche containing ___ frames on each page in accompanying envelope.
7. Nucleotide and/or Amino Acid Sequence Submission
(if applicable, all necessary)
a. ☐ Computer Readable Copy
b. ☐ Paper Copy (identical to computer copy)
c. ☐ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

8. ☐ Assignment Papers (cover sheet & documents) ___ pages
9. ☐ 37 CFR §3.73(b) Statement ☒ Power of Attorney
(combined when there is an Assignee) with Patent Declaration above.)
10. ☐ English Translation Document (if applicable)
11. ☒ Information Disclosure Statement (IDS) & ☒ PTO-1449 ☐ ___Copies of IDS Citations/References
12. ☐ Preliminary Amendment ___ pages
13. ☒ Return Receipt Postcard (MPEP 503)
(should be specifically itemized)
14. Small Entity Status
☐ Small Entity Statement Enclosed ___ pages
☒ Statement filed in prior application; and status still proper and desired
☐ Is no longer claimed.
15. ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
16. ☒ Other:
☒ Copy of Petition for Extension of Time filed in parent appln.;
☒ Submission of Formal Drawings

17. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information and a preliminary amendment:
☒ Continuation ☐ Divisional of prior application No. 09/263,148
Filed on March 5, 1999, entitled: Graphical User Interface Engine For Embedded Systems.
PRIOR APPLICATION INFORMATION: *Examiner* H. Jones *Group Art Unit* 2763

18. CORRESPONDENCE ADDRESS

☐ Customer Number or Bar Code Label or ☒ Correspondence address below

Name	**Norman R. Klivans**			Reg. No. 33,003	
Attorneys for Applicant	**Skjerven Morrill MacPherson LLP**				
Address	25 Metro Drive, Suite 700				
City	San Jose	State	CA	Zip Code	95110
Country:	United States	Telephone	(408) 453-9200	Fax	(408) 453-7979

686009 v1

Please type a plus sign (+) inside this box ⇒ [+]

19. Fee calculations.

CLAIMS (Number Filed)	(1) FOR	(2)		(3) NUMBER EXTRA		(4) RATE		(5) CALCULATIONS
17	TOTAL CLAIMS (37 CFR 1.16(c))	-20	=	0	x	$18	=	$ 0.00
3	INDEPENDENT CLAIMS (37 CFR 1.16(b))	-3	=	0	x	$80	=	$ 0.00
☐	MULTIPLE DEPENDENT CLAIMS (if applicable) (37 CFR 1.18(d))				+	$260.00	=	
	BASIC FEE (37 CFR 1.16(a))						=	$ 710.00
	Total of above Calculations						=	$ 710.00
	Reduction by 50% for filing by small entity (Note 31 CFR 1.9, 1.27, 1.28).						=	$ 355.00
	Fee for Extension of Time (1 month)						=	$ 55.00
	TOTAL						=	$ 410.00

20. FEES: The Commissioner is hereby authorized to credit overpayments or charge the following fees to Deposit Account No. **19-2386**:

a. ☒ Fees required under 37 CFR 1.16. (U.S. Application Filing Fees)
b. ☒ Fees required under 37 CFR 1.17. (Conditional Extension of Time Fees)
c. ☐ Fees required under 37 CFR 1.18. (Patent Issue Fees)

21. ☐ Other: _____

NOTE: **The prior application's correspondence address will carry over to this UPA UNLESS a new correspondence address is provided below.**

22. NEW CORRESPONDENCE ADDRESS

☐ Customer Number or Bar Code Label _____

☐ New correspondence address below

NAME					
ADDRESS					
CITY		STATE		ZIP CODE	
COUNTRY		TELEPHONE		FAX	

23. SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Skjerven Morrill MacPherson LLP
25 Metro Drive, Suite 700
San Jose, CA 95110
Tel. (408) 453-9200 Fax. (408) 453-7979

Date:	October 20, 2000	
Name	Norman R. Klivans	Reg. No. 33,003
Signature	Norman R. Klivans	
Express Mail Label No.	EL 702 144 275 US	

GRAPHICAL USER INTERFACE ENGINE FOR EMBEDDED SYSTEMS

Kenneth J. Klask

FIELD OF THE INVENTION

This invention relates to computer systems and more specifically to embedded systems, i.e. other than general purpose programmable computers.

BACKGROUND

Embedded systems are well known; this refers to microprocessors and microcontrollers (hereinafter generically referred to as microprocessors) used in devices other than general purpose computers. For instance many household appliances (such as microwave ovens) have embedded microprocessors which control operation of the appliance. The microprocessor typically accepts user input, for instance from the keypad of the microwave oven, and controls operation of the microwave oven, for instance the level of heating and duration of cooking. The embedded microprocessor also controls the device display which in a microwave oven is a small LCD (liquid crystal display). That is, the intelligence of such appliances resides in the embedded microprocessor, which interfaces to the human user. Typically this is done through firmware, i.e.

computer software executed by the embedded microprocessor and stored in a memory associated with, or a part of, the microprocessor. In addition to executing the software to interact with the controlled device, the embedded microprocessor also accepts and decodes input from the human user, for instance via the keypad, as well as provides visual feedback on the display by providing text and/or graphic information to a display controller which in turn drives the LCD panel.

As shown in the block diagram of Fig. 1, the embedded microprocessor 10 (in the drawing designated by the alternative terminology "microcontroller") is a commercially available device, for instance an 8 or 16-bit microcontroller of the type available from a number of vendors. This embedded microprocessor conventionally includes, in addition to its logic circuitry, storage such as ROM (read only memory) which holds what is called firmware 12, which is a type of computer software, and also conventional RAM (random access memory) which is not shown. Firmware 12 performs the indicated functions of application flow, device control (of the controlled device of which the embedded microprocessor is a part) reaction to user input, and the capability to draw pixels to the display

controller 24 frame buffer 30.

As shown, the microprocessor 10 is coupled to a user input device 14, e.g. a keypad, an infrared remote controller such as used on television sets, or a touch screen input device. The associated controlled device (not shown) is, for instance, an appliance such as a microwave oven, washing machine, or automobile system, or a scientific instrument, or a machine tool, and is connected conventionally to microprocessor 10. It is to be appreciated that the lines connecting the blocks in Fig. 1 represent buses, that is, parallel multiline connections. The embedded microprocessor 10 supplies input (commands) from the human user via the user input device 14 to control the controlled device and gives user indications on the display 20. Display 20 is driven via conventional pixel drivers/video circuitry 22. The user input device 14, of course, does not directly affect the controlled device, nor does it directly control the display processor 20. Instead, the embedded microprocessor 10 accepts and decodes the user input from the user input device 14, then controls the controlled device and provides information to the user on display 20. Similarly, the display device 20 does not directly display information from the user input device 14, nor the controlled device; instead it

only displays information provided to it by the embedded microprocessor 10. This display takes place via the display controller 24, which is often a separate, commercially available, integrated circuit. Display controller 24 includes several well known elements which are the microcontroller (microprocessor) bus interface 28, which drives the frame buffer 30 and the associated LCD/video interface 34. As shown, the display device is for instance an LCD (liquid crystal display), VFD (vacuum fluorescent display), CRT (cathode ray tube), etc.

The Fig. 1 system is well known and has been in use for many years. It is generally suitable for a high volume production products such as household appliances where manufacturing (parts) cost is important and nonrecurring engineering charges for developing software are relatively less important. The reason for this is that the firmware executed by the microprocessor 10 must be customized for each class of controlled device, as well as for the user input device 14 and the display 20. This requires a substantial amount of software engineering effort. However, this approach is less well adapted for non-mass-produced products such as industrial control systems, or limited production products where the software engineering

costs are relatively more important than the costs of the integrated circuits. Also, even for mass produced products that are subject to frequent changes in the firmware to be executed by the embedded microprocessor 10, the costs of changing the firmware are high and the Fig. 1 approach is relatively expensive and inefficient. Hence, this approach has significant drawbacks in terms of development time and engineering cost.

SUMMARY

In accordance with this invention, a control system, for instance an embedded control system for controlling a device, operates such that the burden of accepting human user (or machine) input and providing information (output) to a human user or a machine via, e.g., a display is shifted from the embedded microprocessor to a second processor. The second processor, designated here a "hypertext" processor, is e.g. a microprocessor, microcontroller, or similar structure capable of processing a hypertext markup language document, as explained below. The embedded control system controls and/or monitors the controlled device and is application specific, unlike for instance a personal computer which can run any application

program. The display controller of Fig. 1 is effectively eliminated and its functions instead associated with the hypertext processor. Both the user (or machine) input device and the display (or other output device) are coupled to the hypertext processor and not to the embedded microprocessor. The hypertext processor is a second, e.g., microprocessor which may be on a chip separate from the embedded microprocessor.

The hypertext processor determines what operations to take upon receipt of, e.g., user input, for instance from a connected keypad. The hypertext processor performs actions described in the hypertext markup language document and commands the embedded microprocessor to act on the controlled device and to update its internal shared variables. The hypertext processor also updates the display as a function of the shared variables internal to the embedded microprocessor. The user interface software (code) is not resident in the hypertext processor, nor is it executed/interpreted by the embedded microprocessor. Instead, a (hypertext) document describing the user interface is external to the hypertext processor, and resident in the memory space of the embedded microcontroller or in a serial memory device (i.e. serial EEPROM, FLASH ROM, smart card, etc.). This

hypertext document describing the user interface is provided ("served") to the hypertext processor at the request of the hypertext processor. Thus the user interface is actually executed by the hypertext processor even though it does not permanently reside there.

In one embodiment, the user interface document is encoded in a particular hypertext markup language (HTML) called here μHTML. The generic name "Hypertext Markup Language" refers to:

> Hypertext - A method for providing links within and between documents; popularized by multimedia authoring systems which used the hypertext concept to link the content of a text document to other documents encoded in certain multimedia formats.
>
> Markup Language - A method for embedding special control codes (TAGS) that describe the structure as well as the behavior of a document.

Like conventional HTML, μHTML files ("documents") contain both control information (markup tags) and content (ASCII text), which together describe the appearance and content of a user interface. In addition, both markup languages provide capability to reference resources external to the document. Compared to conventional HTML, μHTML is smaller, easier to interpret, and defines a special library of GUI (graphical user interface) objects, data processing objects suitable for pipelining, and other system

utilities common to embedded systems software. One key feature of μHTML is its ability to describe the interface to resources distributed among networked embedded subsystems and to link the data of these resources to the functions of a host processor.

In order to make μHTML easy to parse, it is headed by a directory of pointers to each tag. To make it compact, each tag is represented by a single byte (hereinafter referred to as an opcode). Following each opcode is a unique set of binary properties data, such as X-Y coordinate information, pointers to other resources, and other properties. There is a unique opcode for each object in the GUI object library. These objects are, e.g., push buttons, pop-up lists, and other sorts of visual (or aural) indicators. There are also opcodes for objects that contain methods to process or redirect data to and from other objects or external resources, e.g., an object referencing a variable from "external resource 0" may sample the variable data every 100 mS, and route the results to another object referencing a variable from "external resource 1". Each library object opcode is followed immediately by a data structure unique to the object. The data contained in the data structure is specific to the instance of the library object. In this way, the

memory allocated for each instance of all used objects is statically allocated in the memory buffering the μHTML document. When external resources are referenced, a data structure is provided to describe the format of the messages required to provide access to the external resource. For instance, to read a variable associated with an external device, the data structure describes a "Get" command and a "Return" response. Typically the Get command contains an identification to some external device and an index into a lookup table on the external device that provides references to variables, functions or files. In addition to the external device identification and lookup table index, the Return response also contains the data requested.

In one embodiment this user interface hypertext document is developed using conventional internet web page development tools of the type commercially available; this is not limiting. User interface objects are simulated in one embodiment with JAVA applets that correspond to objects in the GUI object library. The simulated GUI objects are referenced from within the conventional HTML document by using the same standard tags used to reference any conventional JAVA applet. Standard HTML tags are also used to format the

display content and to point to resources resident to devices external to the hypertext processor.

The user interface document can then be viewed on a conventional web browser, for system development purposes. (Of course this has little to do with the actual user operation of the controlled device but is part of its user interface design and development.) This HTML/JAVA web page can then be converted (pre-compiled) to a more compact μHTML format by a compiler designed specifically to: (1) remove the conventional HTML tags and replace them with a corresponding μHTML opcode; (2) convert the attributes strings of the HTML tags to a binary structure appropriate for the μHTML opcode; (3) replace references to all JAVA applets and parameters with a corresponding opcode and object data; (4) reformat and add additional data to simplify parsing and execution by the hypertext processor, and (5) resolve references to resources external to the hypertext processor (i.e. executable code or variable data resident to an external embedded microprocessor, storage in an external serial memory device, I/O functions of an external serial I/O device, etc.). This is only illustrative of development of a system in accordance with this invention.

Moreover, the present invention is directed to

more than a user interface processor. It is additionally directed to use of a hypertext markup language to provide program flow and structure while linking together resources distributed among embedded subsystems, even if the subsystems do not have user interfaces. That is, the invention more broadly contemplates a dedicated processor programmed with a hypertext markup language rather than with conventional application code.

BRIEF DESCRIPTION OF THE FIGURES

Fig. 1 shows a prior art embedded control system for a controlled device.

Fig. 2 shows an embedded control system in accordance with this invention.

Fig. 3 shows a more detailed diagram of the markup language processor of Fig. 2.

Fig. 4 shows an HTML file and associated request handler in accordance with the invention.

Fig. 5 shows the relationship between the HTML source file of Fig. 4 and a version compiled to μHTML.

DETAILED DESCRIPTION

Fig. 2 shows a block diagram of a control system for a controlled device in accordance with this

invention. Blocks similar to those of Fig. 1 have identical reference numbers. In Fig. 2, the display controller 24 of Fig. 1 is replaced by a second hypertext processor 40 which may be (not necessarily) a single integrated circuit and which is an intelligent device, unlike the display controller 24. Thus in the Fig. 2 structure there are two intelligent devices (processors), one of which is the hypertext processor 40 and the second of which is, e.g., the embedded microprocessor (or other device) of which several are shown labeled 42a etc. The hypertext processor 40 interfaces both to the user input device 14 and to the display elements 20, 22. Any networked device such as 42c or 42d that contains storage for the user interface (hypertext) document may serve (provide) the user interface document to the markup language processor 40. Any networked I/O device such as 42a, 42b, or 42d that acts upon a controlled or monitored device 29 may have resources that are referenced by the user interface document(s). "Networked" here refers to device connectivity using standard protocols. It includes both "intra-product" networking (connecting several devices within one enclosure) and "inter-product" networking (connecting devices each in its own enclosure.)

Fig. 2 shows different types of devices optionally connected by a conventional network 46 to markup processor 40. These connected devices include embedded microcontroller 42a, serial I/O (input/output) device 42b, μHTML storage device 42c, and embedded microcontroller GUI server 42d including its own μHTML storage. Of course other connection arrangements are possible with any number or combination of devices or networks connected to the markup language processor 40 as long as there is at least one device, e.g., 42c capable of storing the μHTML document(s). Also, because a single μHTML document may contain links to the resources of different devices on the network, it is not necessary for every device on network 46 to contain storage for μHTML documents.

Although Fig. 2 shows only one controlled device 29 connected to a plurality of devices, there may be one or more such controlled devices that may be controlled (or monitored) by one or more of the networked I/O devices 42a, etc. In addition, the networked I/O 42a, etc. devices may or may not be located in the same physical enclosure. For example, the components of a microwave oven may be networked in the same physical enclosure. However, the components of a home entertainment system (e.g., surround sound

receiver/amplifier, VCR, CD/DVD player) may all be networked to a hypertext processor, e.g. in a television set, but each housed in its own physical enclosures.

Also, while the various blocks 30, 40, 20, 22, and 42a, 42b etc. of Fig. 2 in one embodiment are separate integrated circuits, the partitioning amongst the various integrated circuits may be otherwise, for instance, all of the Fig. 2 system may be on a single integrated circuit with the possible exception of the user input device 14, controlled device 29, and display 20. The partitioning of the depicted blocks amongst various integrated circuits is not critical to this invention.

The following describes each functional block of the hypertext processor 40 of Fig. 2:

Network controller 58 formats and transmits all bytes of data queued by the μHTML processor 60 via the network 46. It also decodes any data received from the network 46 and places it in a queue to be processed by the μHTML processor 60.

User input decoder 62 detects and decodes input from user input device 14 which is, e.g., a keypad, touch screen, voice command decoder or IR (infrared) remote device. Decoder 62 places data describing a

user input event into a queue to be processed by the µHTML processor 60.

µHTML processor 60 operates on data stored in µHTML buffer 64 to reflect events queued from the user input decoder 62 and network controller 58. Processor 60 is also responsible for generating and queuing events for the network controller 58 in response to system or user events that are linked to such events by the data in the µHTML buffer 64.

µHTML buffer 64 is RAM (random access memory) storage for a complete µHTML document describing all objects to be rendered to the display device 20. Each object contained in the µHTML document may also contain references to other network resources. Buffer 64 is only written to and modified by the µHTML processor 60 in response to user input events, system events or events generated in response to network messages. It is read by both the rendering engine 52 and the µHTML processor 60. µHTML buffer 64 is a section of RAM 72 accessible only by the microprocessor 68(see Fig. 3).

The rendering engine 52 only reads the graphic information for each UI object as required to properly draw the user interface to the frame buffer 30. The µHTML processor 60 reads the information required to generate system or network events in response to other

events related to each UI object.

Rendering engine 52 draws all displayable user interface objects to the frame buffer 30 as described by the data stored in the µHTML buffer 64. It refreshes each UI object when marked as "dirty" in the µHTML buffer 64 by the µHTML processor 60. Rendering engine 52 is firmware executed by microprocessor 68 and stored in ROM 70(see Fig. 3). Each µHTML object contains code to render all views of the object.

Frame buffer 30 is RAM storage that contains the data for each pixel of the entire displayed page. It is written to by the rendering engine 52 as it draws the user interface on display 20. It is read from by the pixel serializer 36 as it converts the pixel information to signals appropriate to drive the physical display 20. Frame buffer 30 of Fig. 2 is a section of RAM 72 (see Fig. 3) accessible by microprocessor 68 (see Fig. 3) and the pixel serializer 36.

Pixel serializer 36 generates a continuous pixel stream in a format compatible with a specific commercially available physical display 20. As an example, when interfacing to an LCD panel (display 20), the pixel serializer collects and formats each line of pixel data from the frame buffer 30 and synchronizes it

with the conventional display driver pixel clock, frame pulse and line pulse signals. The pixel clock signal clocks the pixel data into the display drivers' internal shift register. The line pulse signal indicates the end of a display line while the frame pulse signal marks the first line of the displayed page.

The Fig. 2 structure advantageously allows use of commercially available internet web page authoring tools (such as HTML) to use "drag and drop" graphic user interface authoring for development of microprocessor based embedded systems. Also, it allows a simple and consistent serial interface via network controller 58 to devices 42a, 42b, etc. regardless of the configuration of the display 20. In other words, the intelligence for control of the display 20 is provided in the processor 40 and need not be coded in the embedded microprocessor 42a software.

This eliminates the conventional programming, for example in assembler or C, required to implement graphical user interface objects that are linked to the variables and functions of the embedded microprocessor 10 such as is required in the prior art system of Fig. 1. It also allows development of the program flow by the non-software engineers who typically specify the

application for the controlled device 29 of Fig. 2 and thereby understand the application and user interaction, but not perhaps firmware programming. This allows quicker and more accurate program development while freeing up the experienced firmware developers to concentrate on the technical program and also yielding better partitioning of a development project into smaller more manageable chunks that may be developed in parallel.

Fig. 3 shows a "hardware" oriented block diagram of the hypertext processor 40 of Fig. 2. Processor 40 connects to one of the embedded devices 42a etc. In this case, the protocol engine 58 of Fig. 2 is shown as queued serial interface 58' which is, for instance, a UART/SPI/I^2C interface. These are examples of industry standard interfaces suitable for the "intra-product" networking described above. SPI (Serial Peripheral Interface) is a popular synchronous serial communication scheme for networking of integrated circuits contained in embedded systems. It was designed by Motorola and popularized by MAXIM, Harris, SanDisk, and others. It is supported by many microcontrollers and serial I/O devices such as A/D and D/A converters, solenoid drivers, digital potentiometers, real time clocks, EEPROM, FLASH ROM,

among many others. I^2C-Bus (Inter-IC Bus) is another popular synchronous serial network architecture popularized by Philips and is simpler, but slower than SPI. Like SPI, many serial I/O and storage functions are available. However, many more consumer product functions are available, i.e. television and stereo building blocks. Examples of suitable interfaces for protocol engine 58' for "interproduct" networks are IEEE-1394, USB, or Ethernet. In conjunction with appropriate firmware executed by the microprocessor 68 and stored in ROM 70 protocol engine 58 services interrupts generated by the connected devices and manages queues.

The user input decoder 62 is shown in Fig. 3 as a keypad scan decoder 62' which connects to a keypad 14. In conjunction with appropriate firmware executed by the microprocessor 68 and stored in ROM (read only memory) 70, decoder 62 services interrupts generated by the connected devices and manages queues. The remaining blocks in Fig. 3 support the other functions of markup language processor 40 of Fig. 2. This is accomplished in terms of circuitry by microprocessor "core" (this is the microprocessor without the supporting memory, etc.) 68 which in turn is connected to a standard bus 76 which interfaces as shown to the

other blocks within processor 40. Typically, the entire processor 40 of Fig. 3 would be a single integrated circuit.

µHTML Processor 60 of Fig. 2 in Fig. 3 is firmware executed by microprocessor 68 and stored in ROM 70. In addition to routines to service interrupts, handle events and manage RAM 72 based queues 78 and buffers, this also contains a library of routines that operate on and according to the specific data structures of each µHTML object. These objects may contain, but are not limited to, user interface objects, data processing objects and operating system objects. The data for each instance of an object is contained in the µHTML document buffered in the RAM 72 area called the µHTML buffer 64. Each µHTML object in the library 84 in ROM 70 contains code to (1) access and modify the data defining the instance of the object (from µHTML buffer 64), (2) render all views of the object to RAM frame buffer 30, (3) respond to events related to the object and (4) queue messages to be sent to other network resources.

The structures in Fig. 3 include (in ROM 70) main program storage 88 and event handlers 90 and (in RAM 72) stack 96 and heap 98. Pixel serializer 36 of Fig. 2 is depicted as hardware (circuitry) in Fig. 3.

The block diagrams of Figs. 2 and 3 are descriptive of a range of structures which may be implemented in various combinations of dedicated hardware (circuitry) and software (computer code) executed by various types of processors. The particular partitioning between hardware and software disclosed herein is not intended to be limiting.

Fig. 4 illustrates an example of an application used in accordance with this invention. Specifically, the central portion of Fig. 4, which is the text 86, is an HTML file, that is a hypertext markup language document which links display items of an LCD display 88 to resources of an embedded microprocessor. The various lines of text in 86 contain either: (1) text to be displayed such as "Two Variables" or "LED 0", or (2) markup tags (enclosed between < and >) to reference GUI object library components and link them to resources external to the HTML document and markup language processor. In this example, the embedded microprocessor resources are accessed through the embedded microprocessor software program 92.

The embedded microprocessor resident resources accessed by program 92 are: two variables in this case containing the values 123 and 321, and two functions that in this case turn on an LED and turn off an LED

attached to the embedded microprocessor. The variables are displayed via IntField objects and accessed by sending the commands in the <PARAM Name = "Send"...> tags. Upon receiving the command to GET a variable, the embedded processor executes the code in 92 to lookup the variable and send the value back via the ackClient routine. The IntField object of the markup language processors GUI Object library parses the response as per the <PARAM Name="Return"...> tag to isolate, format and render the value to the LCD's frame buffer.

Likewise the functions referenced by the <PARAM Name="Send"...> are invoked when the user activates the buttons rendered by the FunctBtn objects.

Associated with this document 86 is embedded request handler 92, shown in the right hand portion of Fig. 4 with lines relating it to the markup in document 86. This handler 92 is resident in an embedded microprocessor such as, with reference to Fig. 2, 42a or 42d to provide access to the resources requested via the network. This code in 92 may be implemented in hardware for example in serial memory devices such as, with reference to Fig. 2, 42c or in serial I/O devices such as 42b. The "client" in the code 92 is a reference to the markup language processor 40. Thus,

while the data described by document 86 is actually interpreted by the markup language processor 40, the code 92 is actually executed by the embedded microprocessor 42a in conjunction therewith.

Fig. 5 shows a repetition of the HTML source file (left side) 86 of Fig. 4 with a compiled μHTML version of same (right side). This compiled μHTML version is much more compact; the lines relate the source file code to its compiled version. In addition, the μHTML is easier to interpret at runtime, because things such as string lengths, tag offsets, X-Y coordinates are computed by the compiler and built into the structure of the document. Of course there is no requirement to use HTML or μHTML or to compile same, however, this provides efficiencies in carrying out one embodiment of the present invention.

Alternatives to use of the μHTML disclosed here are other forms of text documents with control codes used to access resources located elsewhere. Examples of other markup languages are compact HTML, and HDML. Even the old UNIX "troff" is a markup language which was originally designed for page layout.

Memory devices (such as 42c) (Fig. 2) external to the processor 40 are thereby responsible for "hosting" the μHTML and other files. Whether the external device

is another microprocessor 42d, or simply a serial memory device 42c, it reacts to requests from the processor 40 to read or write files. In addition devices 42a etc. connected to the processor 40 may also support requests to read/write variables, invoke functions and provide state information while performing the normal I/O device functionality.

The embedded memory device 42c is thereby responsible for "hosting" the μHTML and other files. It responds to requests from the hypertext processor and keeps track of changes to variables in use by the hypertext processor and executes the controlled device functionality. The hypertext processor is responsible for rendering the graphical user interface to the display. The hypertext processor is also responsible for responding to user input from the user input device by updating display graphics and communicating with external devices to request changes to the values of external variables and to invoke external functions as described by the μHTML document. The hypertext processor is also responsible for responding to changes in the embedded microprocessor variables by updating the display device graphics. Typical requests to the embedded microprocessor by the hypertext processor are: open connection; get file (for instance a μHTML file,

an image graphic file or a script); call up functions; get a value of the variable; send value of the variables and obtain status of the embedded microprocessor. "Script" refers here to files that contain code to be executed by the microprocessor portion of the hypertext processor.

This disclosure is illustrative and not limiting; further modifications will be apparent to one skilled in the art in light of this disclosure and are intended to fall within the scope of the appended claims.

CLAIMS:

What is claimed:

1. An embedded control system to control or monitor an associated device, wherein the associated device includes an input/output portion, the embedded control system comprising:

a processor coupled via input/output circuitry to the input/output portion for controlling operation of the associated device; and

a memory associated with the processor, the memory storing one or more documents linked to each other, but not containing executable code, to describe methods to control or monitor the associated device by linking functions executable by and local to the processor with descriptions of the interactions required to access the input/output circuitry;

wherein the processor receives the stored document from the memory and performs the control or monitor methods of the stored documents by executing the functions referenced by the stored documents to operate on the input/output circuitry linked to the functions by the stored documents, thereby to control or monitor operation of the

input/output portion.

2. The system of Claim 1, further including a second processor coupled between the processor and the input/output circuitry, the second processor including internal resources to facilitate controlling or accessing the input/output circuitry;

wherein the internal resources are linked to functions executable by the first processor via links from the stored documents, thereby to control or monitor operation of the input/output portion.

3. The system of Claim 1, wherein the processor includes a memory to store a copy of the stored document while it executes the linked functions; and the stored document contains an initialized data structure representing the variable data for each function being linked thereto, thereby allocating space in local memory of the processor for each instance of the functions linked from the stored document.

4. The system of Claim 1, wherein the input/output portion includes driver circuitry and a display coupled to the driver circuitry.

5. The system of Claim 1, further comprising an additional processor with an associated memory coupled to the processor.

6. The system of Claim 5, wherein the coupling is via a network.

7. The system of Claim 6, wherein the network is one of an inter-product or an intra-product network.

8. The system of Claim 1, wherein the functions executable by and local to the processor are objects for controlling or monitoring the input/output circuitry coupled to the input/output portion, thereby to control functions of the associated device.

9. The system of Claim 8, wherein the objects include graphical user interface objects.

10. The system of Claim 1, further including stored documents that do not contain executable code, but do contain links to other stored documents that do contain executable code.

11. The system of Claim 1, wherein the input/output portion includes user input decoding circuitry coupled to a user input device.

12. The system of Claim 1, wherein the stored document is a hypertext markup language document.

13. The system of Claim 1, wherein the stored document is compiled from a hypertext markup language document by partitioning the interpretation of the document into a compilation time phase and a run time phase, wherein the compilation phase produces an intermediate document to be stored in the memory and executed by the processor during the run time phase, thereby reducing the burden on the processor for executing the stored document.

14. The system of Claim 2, wherein the second processor is housed in an enclosure separate from an enclosure housing the processor.

15. A method of controlling a device having an associated processor and an input/output portion, comprising the acts of:

storing a document which describes methods to

control or monitor the associated device by linking functions executable by and local to the processor with descriptions of the interactions required to access the input/output portion;

upon a request, transmitting at least a portion of the stored document to the processor; and

executing the transmitted document at the processor, thereby to control operation of the input/output portion.

16. The method of Claim 15, wherein the document is a hypertext markup language document.

17. A method of implementing a user interface for an embedded system with a controlled/monitored device and an associated processor and an input/output portion, comprising the acts of:

developing a document which describes methods to control or monitor the device and interact with a user of the device by linking functions executable by and local to the processor with descriptions of the interactions required to access the input/output portion;

storing the document;

upon a request, transmitting at least a portion of the stored document to the processor; and

executing the transmitted document at the processor, thereby to allow the user to interact with the device.

GRAPHICAL USER INTERFACE ENGINE FOR EMBEDDED SYSTEMS

Kenneth J. Klask

ABSTRACT OF THE DISCLOSURE

In an embedded system, for instance in a household appliance, in addition to the usual embedded microprocessor/microcontroller there is provided another processor which actually executes a user interface HTML document for accepting user input, for instance from a keypad and controlling the display device, for instance an LCD. The embedded microprocessor hosts the user interface document, responds to requests from the other processor, keeps track of changes in variables shared with the other processor, and executes the control device functionality. The other processor renders the graphical user interface to the display and interacts with the user by executing local functions to operate on the memory and i/o resources of the embedded processor as described by the user interface document served to it.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	Kenneth J. Klask		
Assignee:	Amulet Technologies, LLC		
Title:	GRAPHICAL USER INTERFACE ENGINE FOR EMBEDDED SYSTEMS		
Serial No.:	Unknown	Filing Date:	Herewith
Examiner:	Unknown	Group Art Unit:	Unknown
Docket No.:			

San Jose, California

Box PATENT APPLICATION
Attn: Official Draftsperson
COMMISSIONER FOR PATENTS
Washington, D. C. 20231

SUBMISSION OF FORMAL DRAWINGS

Dear Sir:

Applicant submits six (6) sheets of formal drawings, consisting of Figures 1, 2, 3, 4A, 4B and 5, in the above-named application. If there are any questions regarding these drawings, please call the undersigned at .

EXPRESS MAIL LABEL NO:

EL 702 144 275 45

Respectfully submitted,

Norman R. Klivans
Attorney for Applicant
Reg. No. 33,003

LAW OFFICES OF
SKJERVEN MORRILL
MacPHERSON LLP

25 METRO DRIVE
SUITE 700
SAN JOSE, CA 95110
(408) 453-9200
FAX (408) 453-7979

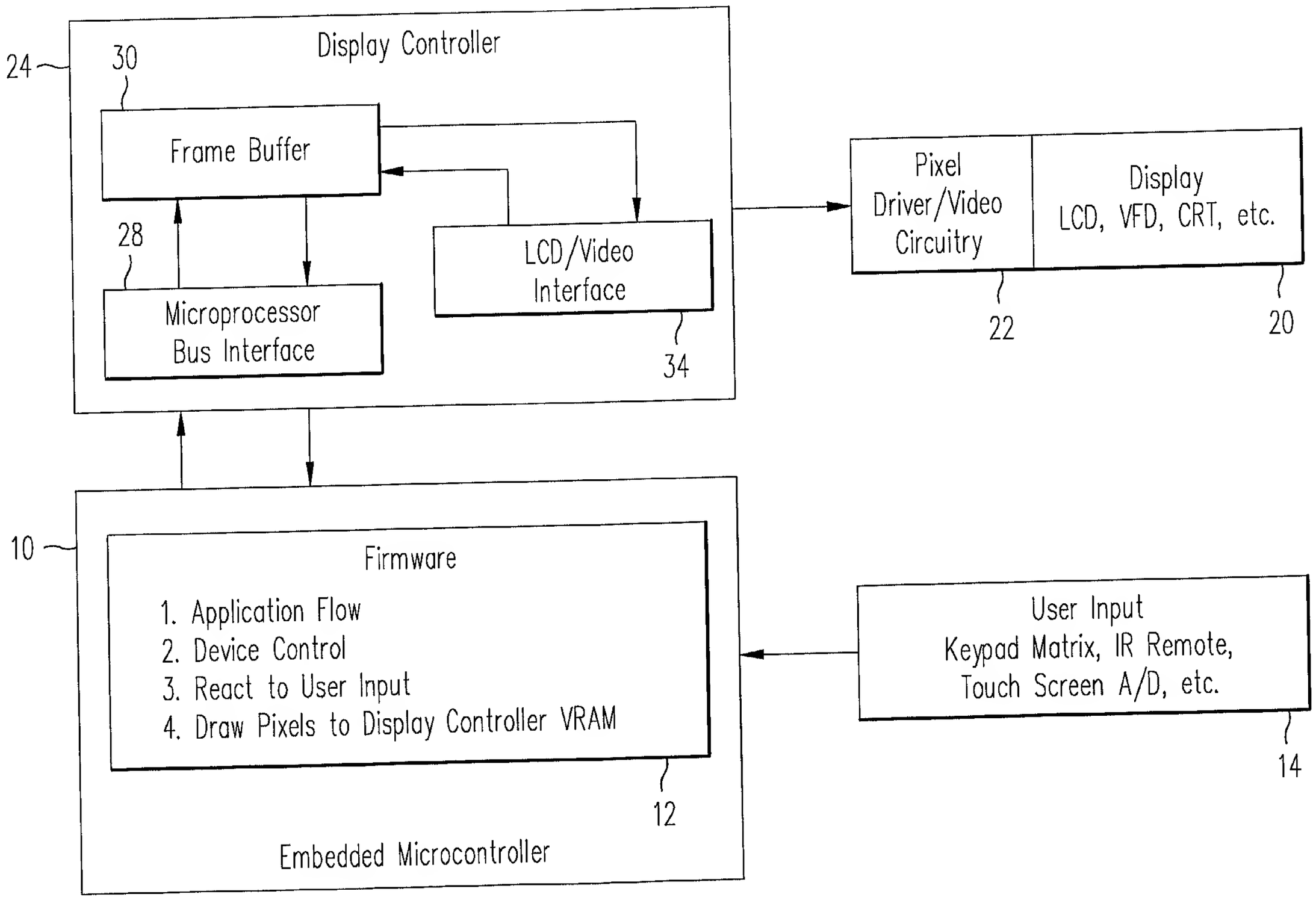


FIG. 1
(Prior Art)

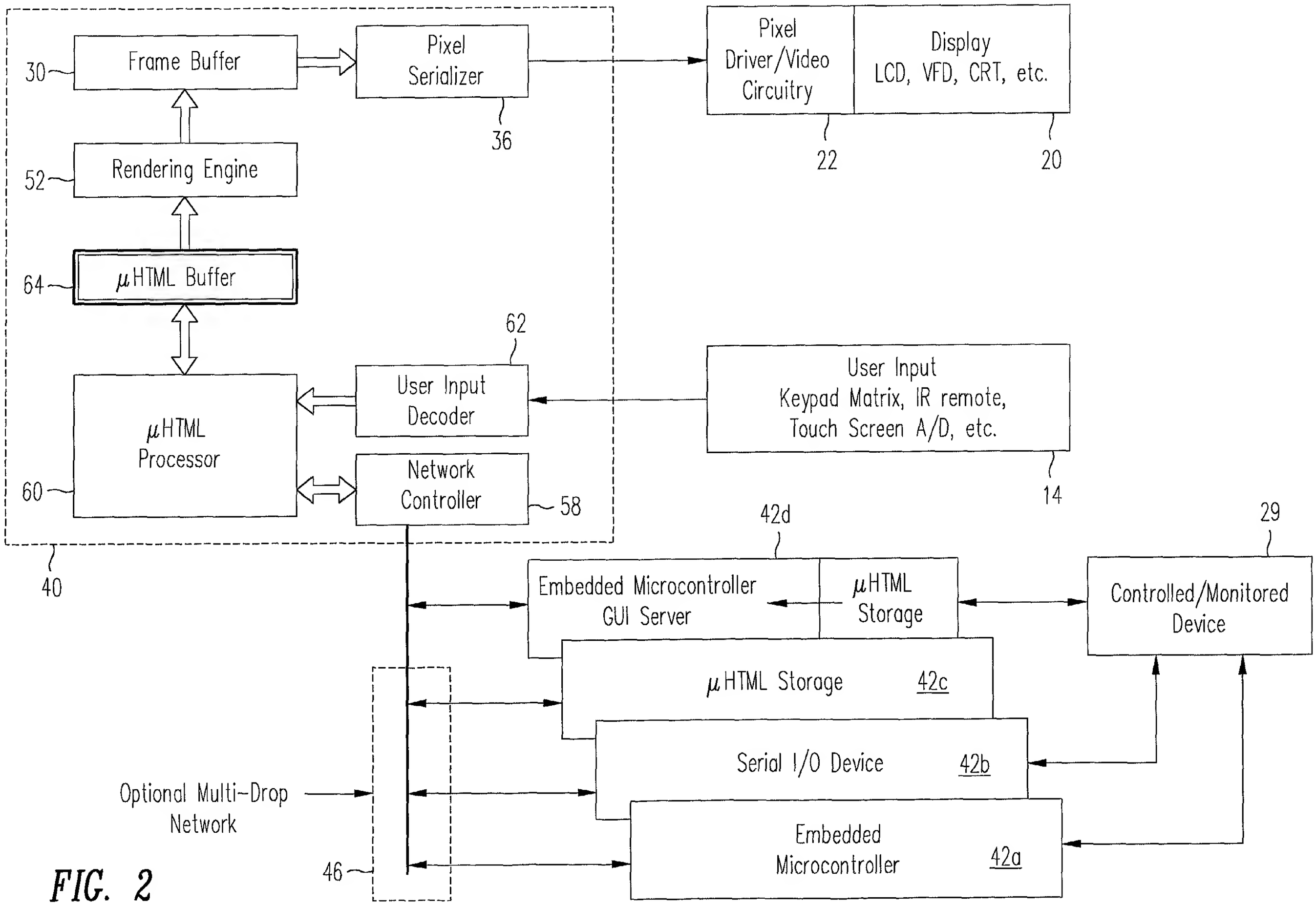

30
Frame Buffer
Pixel Serializer
36
Pixel Driver/Video Circuitry
22
Display LCD, VFD, CRT, etc.
20
52
Rendering Engine
64
μHTML Buffer
62
User Input Decoder
User Input Keypad Matrix, IR remote, Touch Screen A/D, etc.
14
μHTML Processor
60
Network Controller
58
40
42d
Embedded Microcontroller GUI Server
μHTML Storage
29
Controlled/Monitored Device
μHTML Storage 42c
Serial I/O Device 42b
Embedded Microcontroller 42a
Optional Multi-Drop Network
46

FIG. 2

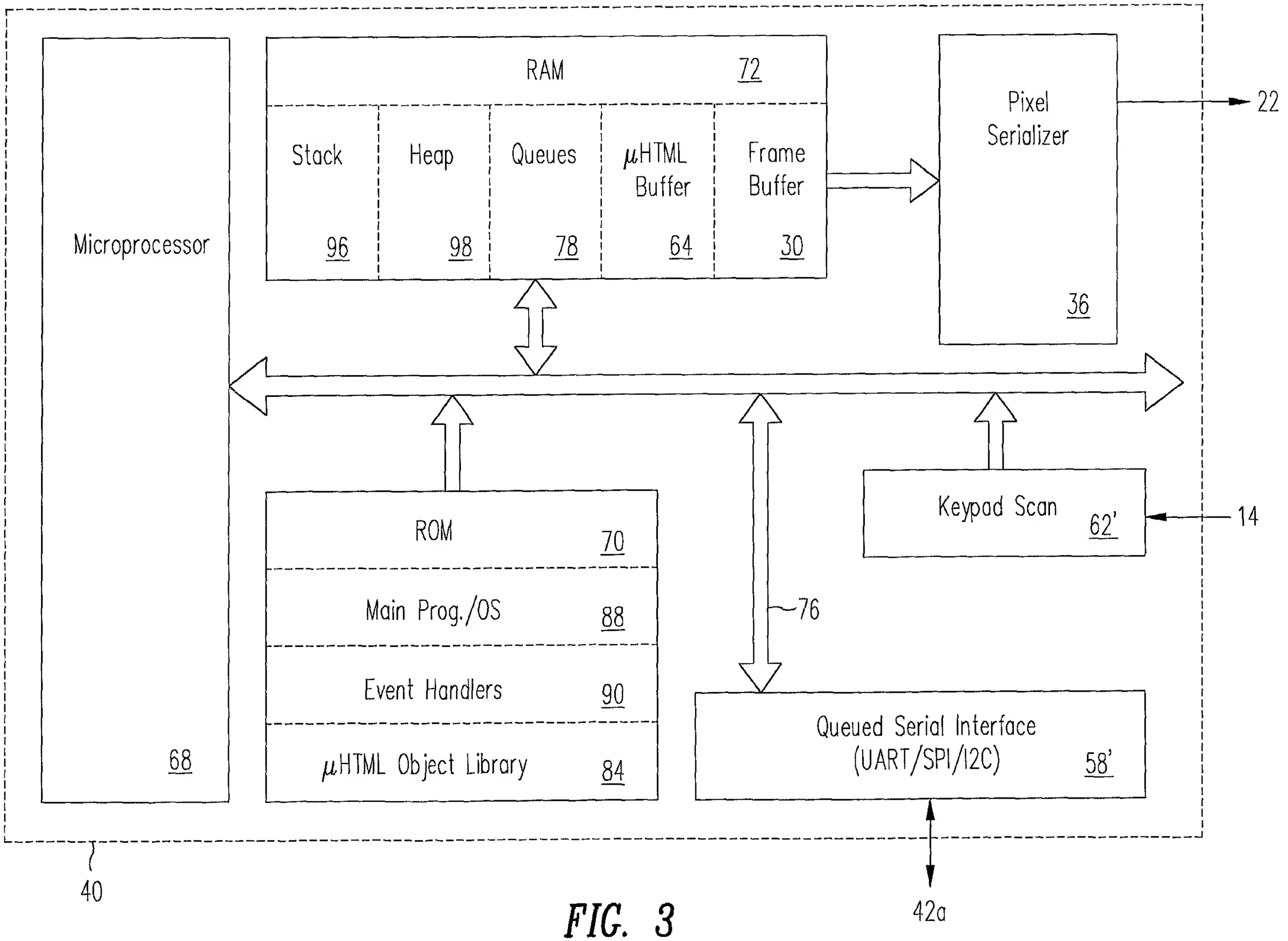
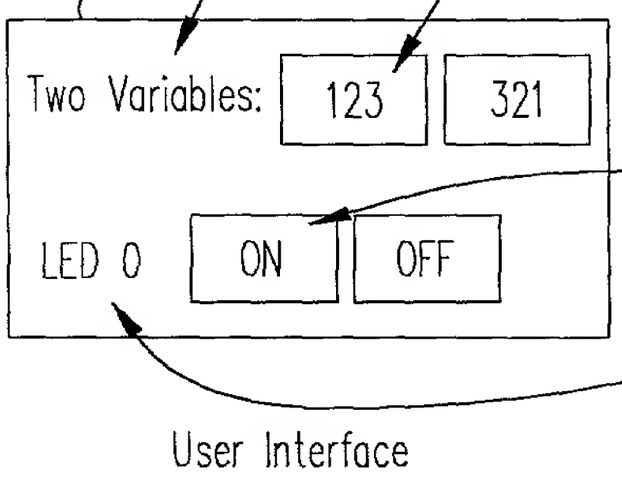

Microprocessor
68
RAM
72
Stack
96
Heap
98
Queues
78
μHTML Buffer
64
Frame Buffer
30
Pixel Serializer
36
22
ROM
70
Main Prog./OS
88
Event Handlers
90
μHTML Object Library
84
Keypad Scan
62'
14
76
Queued Serial Interface (UART/SPI/I2C)
58'
40
42a

FIG. 3

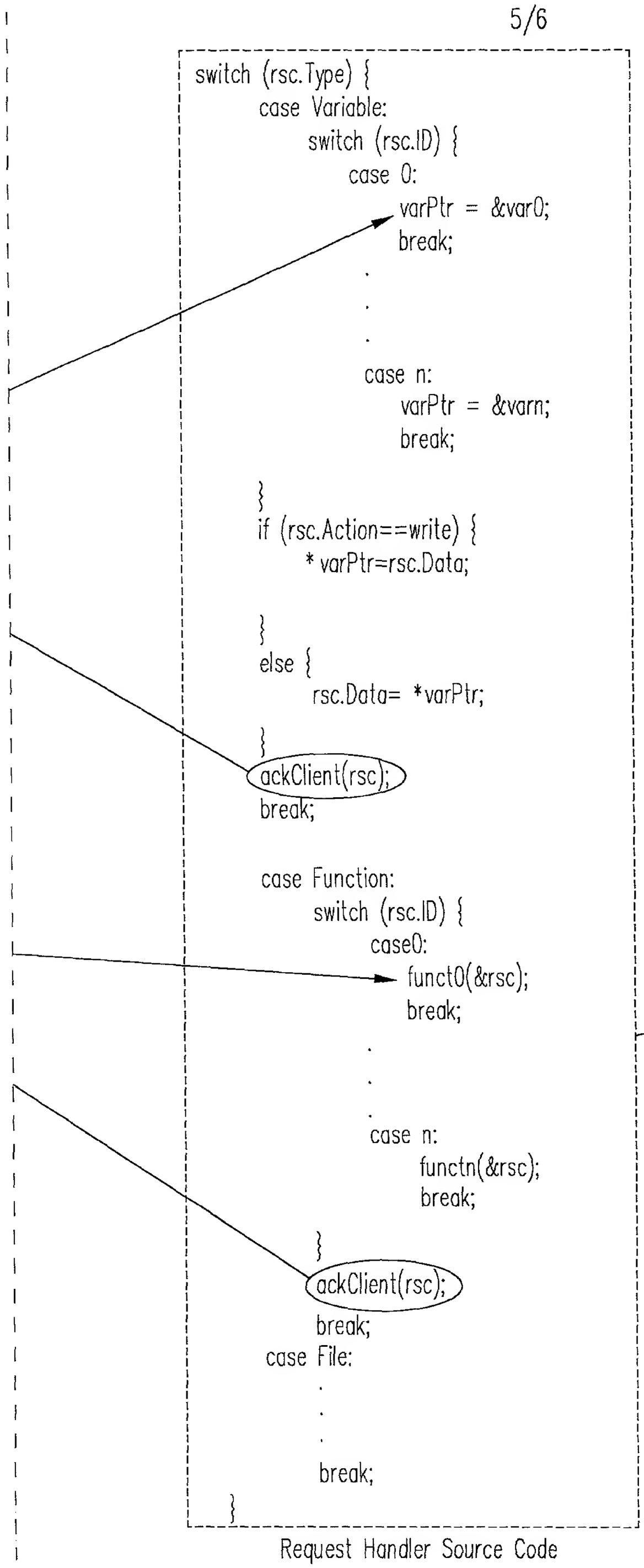


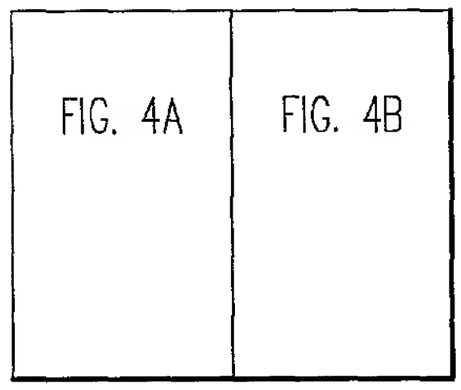

```
<HTML>
<HEAD>
    <TITLE>Amulet--View Variables</TITLE>
</HEAD>
<BODY>
<P>
<TABLE BORDER="0" WIDTH="256" HEIGHT="128">
    <TR>
        <TD WIDTH="40%">Two Variables:</TD>
        <TD WIDTH=30%><APPLET CODE="IntField.class"
            WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
                <PARAM NAME="Send" VALUE="G\x00">
                <PARAM NAME="Return" VALUE="g\x00\x??\r">
            </APPLET>
        </TD>
        <TD WIDTH="30%"><APPLET CODE="IntField.class"
            WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
                <PARAM NAME="Send" VALUE="G\x01">
                <PARAM NAME="Return" VALUE="g\x01\x??\r">
            </APPLET>
        </TD>
    </TR>
    <TR>
        <TD WIDTH="33%">LED 0</TD>
        <TD WIDTH="33%"><APPLET CODE="FunctBtn.class"
            WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
                <PARAM NAME="Send" VALUE="F\x00">
                <PARAM NAME="Return" VALUE="f\x00\r">
                <PARAM NAME="Label" VALUE="ON">
            </APPLET>
        </TD>
        <TD WIDTH="33%"><APPLET CODE="FunctBtn.class"
            WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
                <PARAM NAME="Send" VALUE="F\x01">
                <PARAM NAME="Return" VALUE="f\x01">
                <PARAM NAME="Label" VALUE="OFF">
            </APPLET>
        </TD>
    </TR>
</TABLE>
</BODY>
</HTML>
```

86

HTML Source Code

FIG. 4A

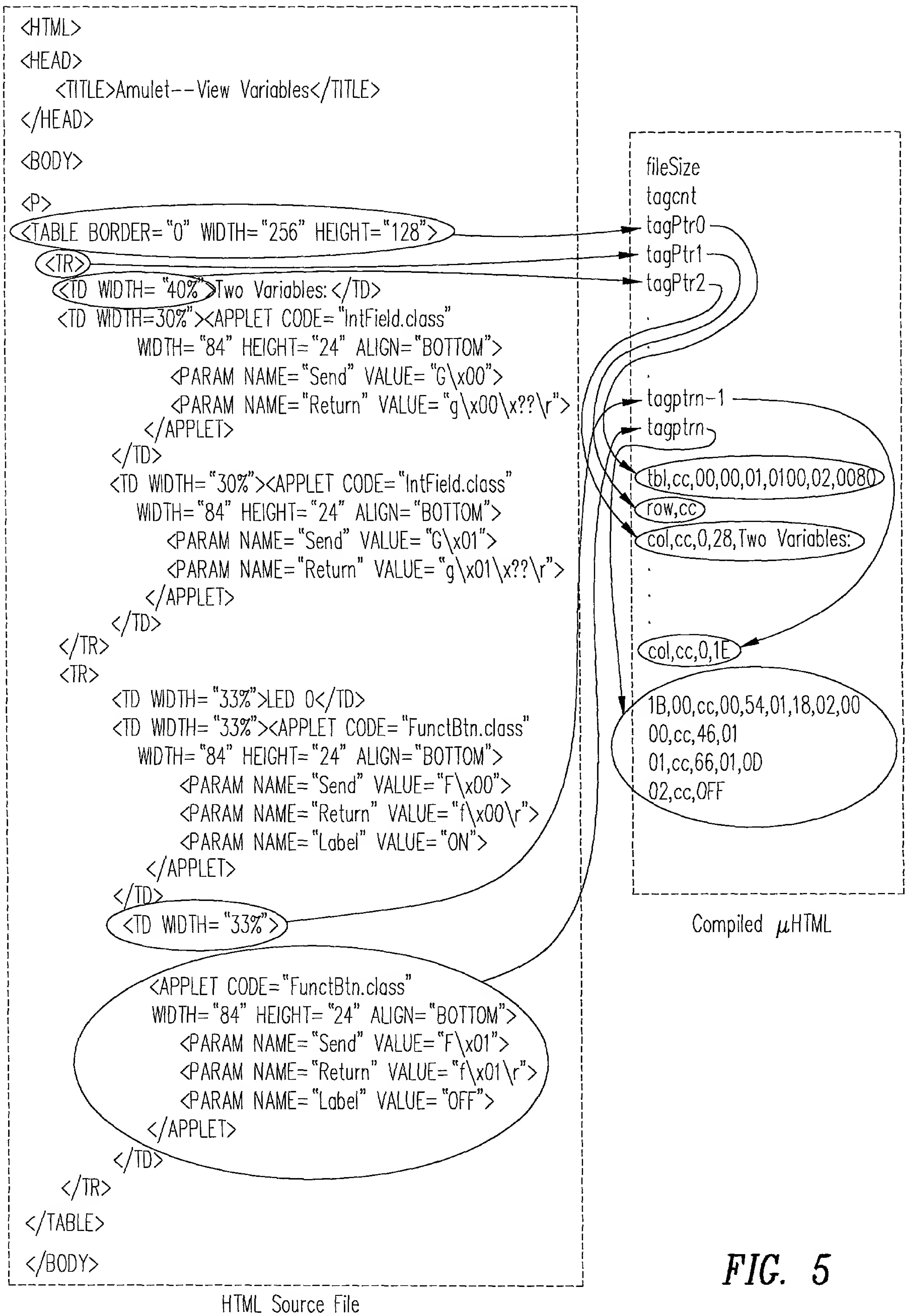




Key To

FIG. 4

FIG. 4B



<HTML>
<HEAD>
<TITLE>Amulet--View Variables</TITLE>
</HEAD>
<BODY>
<P>
<TABLE BORDER="0" WIDTH="256" HEIGHT="128">
<TR>
<TD WIDTH="40%">Two Variables:</TD>
<TD WIDTH=30%"><APPLET CODE="IntField.class"
WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
<PARAM NAME="Send" VALUE="G\x00">
<PARAM NAME="Return" VALUE="g\x00\x??\r">
</APPLET>
</TD>
<TD WIDTH="30%"><APPLET CODE="IntField.class"
WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
<PARAM NAME="Send" VALUE="G\x01">
<PARAM NAME="Return" VALUE="g\x01\x??\r">
</APPLET>
</TD>
</TR>
<TR>
<TD WIDTH="33%">LED 0</TD>
<TD WIDTH="33%"><APPLET CODE="FunctBtn.class"
WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
<PARAM NAME="Send" VALUE="F\x00">
<PARAM NAME="Return" VALUE="f\x00\r">
<PARAM NAME="Label" VALUE="ON">
</APPLET>
</TD>
<TD WIDTH="33%">
<APPLET CODE="FunctBtn.class"
WIDTH="84" HEIGHT="24" ALIGN="BOTTOM">
<PARAM NAME="Send" VALUE="F\x01">
<PARAM NAME="Return" VALUE="f\x01\r">
<PARAM NAME="Label" VALUE="OFF">
</APPLET>
</TD>
</TR>
</TABLE>
</BODY>
HTML Source File
fileSize
tagcnt
tagPtr0
tagPtr1
tagPtr2
tagptrn-1
tagptrn
tbl,cc,00,00,01,0100,02,0080
row,cc
col,cc,0,28,Two Variables:
col,cc,0,1E
1B,00,cc,00,54,01,18,02,00
00,cc,46,01
01,cc,66,01,0D
02,cc,OFF
Compiled µHTML

FIG. 5

Attorney Docket No.: M-6017 US

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled: **GRAPHICAL USER INTERFACE ENGINE FOR EMBEDDED SYSTEMS**

which (check) ☒ is attached hereto.
☐ and is amended by the Preliminary Amendment attached hereto.
☐ was filed on _____ as Application Serial No. _____
☐ and was amended on ___ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
Number	Country	Day/Month/Year Filed	Yes	No
N/A			☐	☐

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Provisional Application Number	Filing Date
N/A	

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal

Regulations, § 1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Application Serial No.	Filing Date	Status (patented, pending, abandoned)
N/A		

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Alan H. MacPherson (24,423); Thomas S. MacDonald (17,774); Brian D. Ogonowsky (31,988); David W. Heid (25,875); Norman R. Klivans (33,003); Edward C. Kwok (33,938); David E. Steuber (25,557); Michael Shenker (34,250); Stephen A. Terrile (32,946); Peter H. Kang (40,350); Ronald J. Meetin (29,089); Ken John Koestner (33,004); Omkar K. Suryadevara (36,320); David T. Millers (37,396); Kent B. Chambers (38,839); Serge J. Hodgson (40,017); Michael P. Adams (34,763); Michael J. Halbert (40,633); Gary J. Edwards (41,008); William B. Tiffany (41,347); James E. Parsons (34,691); Daniel P. Stewart (41,332); Philip W. Woo (39,880); John T. Winburn (26,822); Tom Chen (42,406); Fabio E. Marino (43,339); William W. Holloway (26,182); Elaine H. Lo (41,158); Don C. Lawrence (31,975); Marc R. Ascolese (42,268); Carmen C. Cook (42,433); David G. Dolezal (41,711); Michael P. Noonan (42,038); Roberta P. Saxon (43,087); Bernice Chen (42,403); Mary Jo Bertani (42,321); Dale R. Cook (42,434); and Sam G. Campbell (42,381).

Please address all correspondence and telephone calls to:

Norman R. Klivans
Attorney for Applicant(s)
SKJERVEN, MORRILL, MacPHERSON, FRANKLIN & FRIEL LLP
25 Metro Drive, Suite 700
San Jose, California 95110-1349
Telephone: 408-453-9200
Facsimile: 408-453-7979

I declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all statements made herein are made with the knowledge that whoever, in any matter within the jurisdiction of the Patent and Trademark Office, knowingly and willfully falsifies, conceals, or covers up by any trick, scheme, or device a material fact, or makes any false, fictitious or fraudulent statements or representations, or makes or uses any false writing or document knowing the same to contain any false, fictitious or fraudulent statement or entry, shall be subject to the penalties including fine or imprisonment or both as set forth under 18 U.S.C. 1001, and that violations of this paragraph may jeopardize the validity of the application or this document, or the validity or enforceability of any patent, trademark registration, or certificate resulting therefrom.

Full name of sole inventor: **Kenneth J. Klask**

Inventor's Signature: Kenneth J. Klask Date: 3/2/1999

Residence: San Jose, California

Post Office Address: 4363 Montmorency Court
San Jose, California 95118

Citizenship: U.S.A